```
class Song:
    #Represents a song with the title, artist, and song duration
    def init(self, title, artist, duration):
        #Initializes a song object with the title, artist, and duration
        self.title = title
        self.artist = artist
        self.duration = duration
```

The song class is introduced with the premise of being simple, but details the properties of a song.

```
class Node:
    #Represents a node in a doubly linked list
    def init(self, song):
        #Initializes a node object with the given song
        self.song = song
        self.next = None #Reference to the next node in the list
        self.prev = None #Reference to the previous node in the list
```

The node class holds reference to a song (self.song) and uses pointers to the next and previous nodes. This design allows for efficient traversal and manipulation of the playlist.

```
class Playlist:
    #Represents a playlist containing multiple songs
    def init(self, name):
        #Initializes a playlist object with a name and empty list of songs
        self.name = name
        self.head = None
        self.tail = None

    def add_song(self, song):
        #Adds a song to the playlist
        new_node = Node(song)
        if not self.head:
            self.head = node_node
            self.tail = new_node
        else:
            self.tail.next = new_node
            new_node.prev = self.tail
            self.tail = new_node
```

The playlist class uses a doubly linked list to efficiently manage the songs' order. It includes the attributes 'head' and 'tail' to reference the first and last nodes of the linked list.

```
class Jukebox:
    #Represents the digital jukebox with playlists and playback controls
    def init(self):
        #Initializes the jukebox object with empty playlists and default settings
        self.playlists = []
        self.current_playlist_index = 0
        self.current_song_node = None
        self.is_playing = False
        self.repeat = False
```

The jukebox class is used to manage the playlists and playback controls. We start by initializing an empty list for the playlist, as well as using default settings, as nothing is 'playing'.

```
    def add_playlist(self, playlist):
        #Adds a playlist to the jukebox
        self.playlists.append(playlist)
```

Add_playlist works to add at least one playlist to the jukebox.

```
def play_pause(self):
    #Toggles between playing or pausing the current song
    self.is_playing = not self.is_playing
```

Play_pause allows for pausing a song or playing a song.

```
  def skip_previous(self):
      #Skips to the previous song using doubly linked list traversal
      if self.current_song_node is None:
          return
      #If the current song nodei s not the head of the playlist
      if self.current_song_node.prev:
          self.current_song_node = self.current_song_node.prev
      else:
          #If the current song node is the head of the playlist,
          #move to the previous playlist and set current song node to its tail
          if self.current_playlist_index > 0:
              self.current_playlist_index -= 1
              self.current_song_node = self.playlists[self.current_playlist_index].tail
```

Skip_previous is where we start to employ the doubly linked lists. Using doubly linked lists here allows us to skip backwards to the previous song.

```
def skip_next(self):
    #Skips to the next song using doubly linked list traversal
    if self.current_song_node is None:
        return
    #If the current song node is not the tail of the playlist
    if self.current_song_node.next:
        self.current_song_node = self.current_song_node.next
    else:
        #If the current song node is the tail of the playlist,
        #move to the next playlist and set current song node to its head
        if self.current_playlist_index < len(self.playlists) - 1:
            self.current_playlist_index =+ 1
            self.current_song_node = self.playlists[self.current_playlist_index].head
```

Skip_next allows us to use doubly linked lists to traverse the forwards direction to play the next song

```
def skip_to_first_track_current_playlist(self):
    #Skips to the first track in the current playlist using doubly linked list traversal
    if self.current_playlist_index < len(self.playlists):
        self.current_song_node = self.playlists[self.current_playlist_index].head
```

Skip_to_first_track_current allows us to jump to the first song in the current playlist

```
def skip_to_first_track_previous_playlist(self):
    #Skips to the first track of the previous playlist
    if self.current_song_index == 0 and self.current_playlist_index > 0:
        self.current_playlist_index -= 1
        self.current_song_index = 0
```

Skip_to_first_track_previous_playlist allows us to jump back to the first track of the previous playlist

```
def skip_to_first_track_next_playlist(self):
    #Skips to the first track of the next playlist
    if self.current_song_index == len(self.playlists[self.current_playlist_index].songs) - 1 \
            and self.current_playlist_index < len(self.playlists) - 1:
        self.current_playlist_index += 1
        self.current_song_index = 0
```

Skip_to_first_track_next_playlist allows us to jump forward to the first song on the next playlist

```
def rearrange_playlist_order(self, from_index, to_index):
    #Rearranges the order of playlists
    playlist = self.playlists.pop(from_index)
    self.playlists.insert(to_index, playlist)
```

Rearrange_playlist_order rearranges the order of the playlists using pop and insert

```
  def toggle_repeat(self):
      #Toggles the repeat mode
      self.repeat = not self.repeat
```

Toggle_repeat turns the repeat function on and off

```
#Example

if __name__ == "main":
    #Create songs
    song1 = Song("Song 1", "Artist 1", 180)
    song2 = Song("Song 2", "Artist 2", 240)
    song3 = Song("Song 3", "Artist 3", 200)
```

How the songs get created with their naming attributes

```
  #Create a playlist and add songs to it
  playlist1 = Playlist("Playlist 1")
  playlist1.add_song(song1)
  playlist1.add_song(song2)
```

We can add songs to a playlist by appending the song(number) to the list

```
  #Create another playlist and add a song to it
  playlist2 = Playlist("Playlist 2")
  playlist2.add_song(song3)
```

We have the ability to create another playlist, and add songs using the same method as above

```
#Create a jukebox and add playlists to it
jukebox = Jukebox()
jukebox.add_playlist(playlist1)
jukebox.add_playlist(playlist2)
```

A jukebox can be created by adding playlists to it

```
#Various jukebox functionalities
jukebox.play_pause()  # Toggles play/pause state
jukebox.skip_next()   # Skips to the next song
jukebox.skip_previous()  # Skips to the previous song
jukebox.skip_to_first_track_current_playlist()  # Skips to the first track of the current playlist
```

Finally, we are able to utilize the functions from the Jukebox class, allowing us to play/pause songs, skipping forwards/backwards, and skipping to the first track in the current playlist.